API PENETRATION TESTING COURSE

Mapped to OWASP API Top 10

Day 5 — Defense, Reporting & Final Assessment

Secure API Design Principles & Pentestration Testing Report

Mapped to OWASP API Top 10

ABOUT THE INSTRUCTOR

Muhammad Faqih, S.Kom

Former Security Engineer (Dynamic Application Security Tester) at Bank Rakyat Indonesia, Security Analyst at SGI Asia.

Certifications

OSCP+, eMAPT, eWPTX, eJPT, PT1, CND, Security+

LinkedIn: https://www.linkedin.com/in/siberfaqih

TABLE OF CONTENT

01.

TRAINING OVERVIEW AND OBJECTIVE

TRAINING OVERVIEW

Pelatihan ini membahas prinsip dasar Secure API Design dan Penulisan Laporan Penetration Test.

Peserta akan mempelajari bagaimana merancang, mengamankan, dan memantau API secara efektif melalui kombinasi materi teori dan hands-on lab berbasis kasus nyata.

Fokus utama:

- Membangun API yang aman sesuai standar OWASP API Security Top 10.
- Mengidentifikasi, mengeksploitasi, dan memperbaiki kerentanan umum pada API.
- Menyusun laporan uji penetrasi yang profesional dan mudah dipahami.

TRAINING OBJECTIVE

Setelah mengikuti training ini, peserta diharapkan dapat:

1. Memahami dan menerapkan Secure API Design Principles.
2. Mengonfigurasi Authentication & Authorization yang kuat.
3. Melakukan Validation & Sanitization input untuk mencegah eksploitasi.
4. Mengimplementasikan Data Encryption untuk melindungi data sensitif.
5. Menerapkan Rate Limiting & Throttling untuk mencegah DoS.
6. Menggunakan Logging & Monitoring untuk deteksi insiden keamanan.
7. Menulis penetration testing report yang efektif dan actionable.

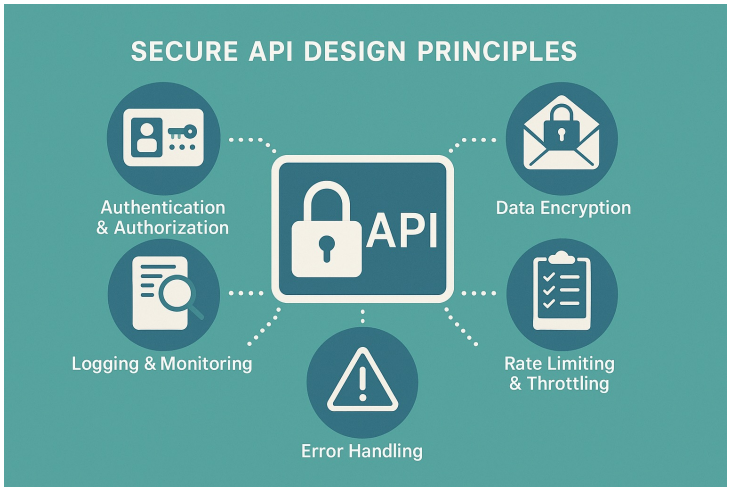
02.

SECURE API DESIGN PRINCIPLES

SECURE API DESIGN PRINCIPLES – (1/4)

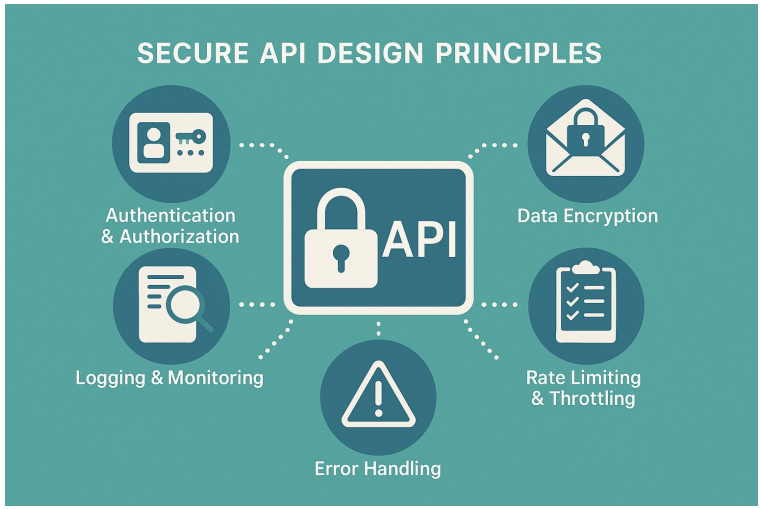
Apa itu API Design Principles?

Secure API Principles adalah sekumpulan prinsip dasar yang digunakan developer dan security engineer untuk mendesain API agar aman sejak awal (security by design). Tujuannya adalah untuk mencegah penyalahgunaan API, kebocoran data, dan serangan seperti injection, brute force, atau privilege escalation.



SECURE API DESIGN PRINCIPLES – (2/4)

Mendesain API yang aman itu artinya kita membuat pintu masuk ke aplikasi (API) supaya data pengguna tetap terlindungi. Jadi, hanya orang yang berhak saja yang bisa masuk, sementara peretas atau pihak yang tidak diinginkan tidak bisa mengakses informasi tersebut.



SECURE API DESIGN PRINCIPLES – (3/4)

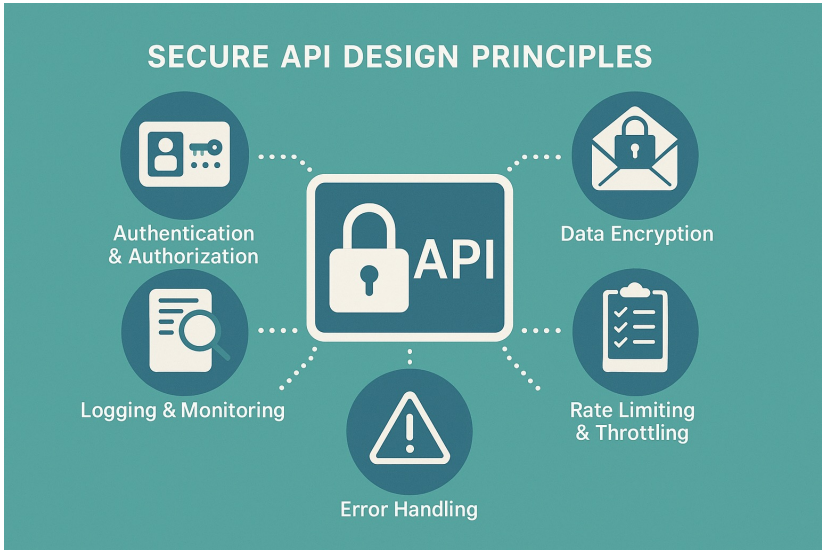
Kenapa Desain API yang Aman Penting?

API itu ibarat jembatan yang menghubungkan satu aplikasi dengan aplikasi lainnya supaya bisa ngobrol dan tukar data. Nah, masalahnya, kalau jembatan ini tidak dibangun dengan aman, ada celah yang bisa dimanfaatkan hacker untuk masuk dan mencuri data penting, misalnya password atau informasi pribadi. Karena itu, API yang baik bukan cuma sekadar berfungsi dengan lancar, tapi juga harus bisa menjaga keamanan data penggunanya.

SECURE API DESIGN PRINCIPLES – (4/4)

Apa saja prinsip-prinsipnya?

1. Authentication & Authorization
2. Validation and Sanitization
3. Data Encryption
4. Rate Limiting & Throttling
5. Error Handling
6. Logging & monitoring API activity



03.

AUTHENTICATION & AUTHORIZATION

AUTHENTICATION & AUTHORIZATION – (1/4)

SGI Asia

Bedanya apa?

AUTHENTICATION

Mengecek identitas

Siapa diri anda?

(WHO)

Contohnya: saat login, kamu masukkan username dan password untuk membuktikan bahwa benar kamu adalah pemilik akun itu.

AUTHORIZATION

Menentukan batasan hak

Apa yang boleh dan tidak boleh dilakukan?

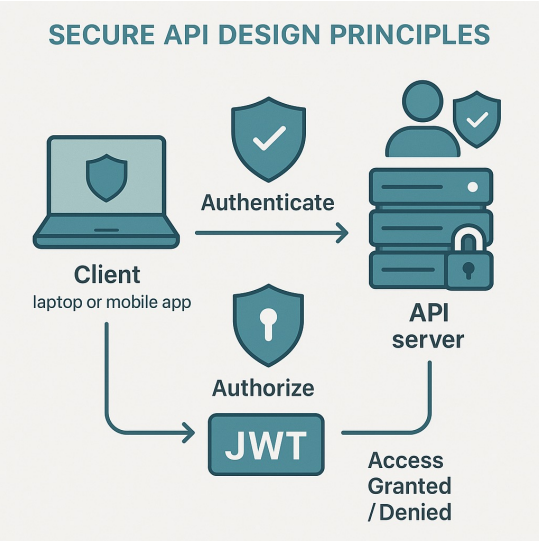
(WHAT)

Contohnya: setelah masuk, mungkin kamu hanya bisa lihat profilmu sendiri, tapi admin bisa lihat semua data pengguna.

AUTHENTICATION & AUTHORIZATION – (2/4)

Harus apa aja?

- **Prinsip:** Pastikan identitas (auth) benar, dan hak akses (authorization) sesuai.
- **Praktik**:
  - Gunakan OAuth2.0 (Authz) / OpenID Connect (Auth) / JWT dengan validasi signature, issuer, dan expiration.
  - Hindari token long-lived; gunakan refresh token dengan rotasi.
  - Terapkan mTLS untuk API internal.
- Kesalahan umum: API hanya pakai API key statis tanpa expiry.



AUTHENTICATION & AUTHORIZATION – (3/4)

Authorization Strategy

RBAC (Role-Based Access Control)

- Akses berdasarkan *role* (Admin, User, Manager)
- Contoh:
  - Admin → CRUD semua data
  - User → hanya baca data sendiri

PBAC (Policy-Based Access Control)

- Aturan dikelola lewat *policy engine* (terpisah dari kode)
- Contoh: gunakan **Open Policy Agent (OPA), AWS IAM Policies, dll**

ABAC (Attribute-Based Access Control)

- Akses berdasarkan atribut & konteks (department, lokasi, waktu, dsb.)
- Contoh: hanya jika department == "IT" dan location == "Jakarta"

AUTHENTICATION & AUTHORIZATION – (4/4)

Contoh vulnerability:

- Alice dapat membuat project baru tanpa harus ada authentication
- Alice dapat melihat project orang lain yang bukan miliknya

OWASP Mapping:

- API2: Broken Authentication → identitas palsu bisa masuk
- API1 & API5: Broken Authorization → pengguna bisa akses atau jalankan hal yang tidak berhak

04.

VALIDATION & SANITIZATION

VALIDATION & SANITIZATION – (1/3)

Bedanya apa?

VALIDATION

Memastikan format input benar

Contohnya: inputan email hanya menerima format email, inputan NIK hanya boleh menerima angka 16 digit, inputan umur harus integer positif.

SANITIZATION

Membersihkan input dari karakter berbahaya

Contohnya: escape karakter berbahaya, misal HTML (< menjadi **<**) sebelum ditampilkan. Memangkas whitespace berlebih.

👉 Validation lebih diutamakan; sanitization dipakai kalau data perlu ditampilkan kembali atau diteruskan ke sistem lain.

VALIDATION & SANITIZATION – (1/3)

SGI Asia

Contoh Vulnerability:

- Seorang attacker dapat melakukan SQL injection
- Seorang attacker dapat memasukkan payload XSS

VALIDATION & SANITIZATION – (3/3)

OWASP Mapping:

- API10 – Unsafe Consumption of APIs

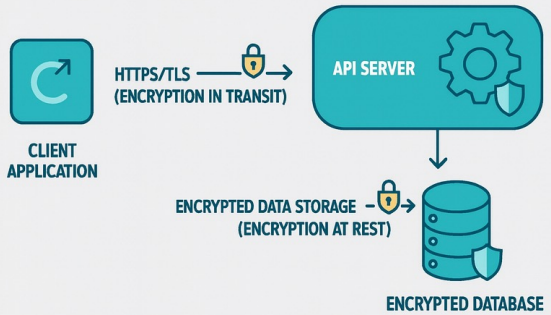
05.

DATA ENCRYPTION

DATA ENCRYPTION – (1/4)

- Enkripsi = mengubah data menjadi kode rahasia.
- Jika data disadap, isinya tetap tidak bisa dibaca tanpa kunci.
- API aman menggunakan enkripsi agar data terlindungi saat dikirim.

Enkripsi itu seperti mengirim surat dalam kotak terkunci. Hanya orang yang punya kuncinya yang bisa membukanya.



DATA ENCRYPTION – (2/4)

IN TRANSIT (Data Saat Dikirim)

Melindungi data yang berpindah antar system

Contohnya: HTTPS (TLS 1.2+), mTLS

At Rest (Data saat disimpan)

Melindungi data di database / file system

Contohnya: AES, RSA, Disk Encryption

DATA ENCRYPTION – (3/4)

Contoh vulnerability:

- Tidak menggunakan HTTPS/TLS saat data in transit
- Tidak melakukan encryption saat data disimpan

DATA ENCRYPTION – (4/4)

OWASP Mapping:

Aspek	OWASP API Category	Fokus Risiko	Contoh Relevansi
Data in transit (HTTPS, TLS, mTLS)	API8: Security Misconfiguration	Salah konfigurasi enkripsi / sertifikat	HTTP tanpa TLS, expired cert
Data at rest (AES, key vault)	API3: Broken Object Property Level Authorization	Data sensitif tidak dilindungi di DB / response	Password plaintext
Key & cert management	API8: Security Misconfiguration	Hardcoded key, insecure cipher	Kunci di source code
Legacy API tanpa enkripsi	API9: Improper Inventory Management	API lama masih aktif tanpa HTTPS	/v1/api masih plaintext

06.

RATE LIMITING & THROTTLING

Rate Limiting & Throttling – (1/6)

Apa itu Rate Limiting?

- Rate Limiting adalah mekanisme untuk membatasi jumlah permintaan (requests) yang dapat dilakukan oleh pengguna atau klien ke suatu API dalam jangka waktu tertentu.

- Contoh:
  - Maksimum 100 request per menit per user.

- Tujuannya: mencegah penyalahgunaan (abuse) seperti brute force login, scraping, atau DDoS yang membebani server.

Rate Limiting & Throttling – (2/6)

Apa itu Throttling?

- Throttling adalah mekanisme untuk mengatur kecepatan pemrosesan request, bukan hanya membatasi jumlah totalnya.

- Kalau rate limiting menolak request berlebih, throttling bisa menunda atau memperlambat respon ketika trafik tinggi.

- Contoh:
  - Server tetap menerima request, tapi menunda respons selama beberapa detik agar sistem tidak overload.

Rate Limiting & Throttling – (3/6)

Mengapa Penting di API Security?

Ancaman	Dampak tanpa Rate Limiting	Dampak yang dicegah
Brute Force Attack	Penyerang bisa menebak password tanpa batas	Akses dibatasi per menit atau per IP
Credential Stuffing	Ribuan kombinasi login dicoba otomatis	Request berlebih akan ditolak
Denial of Service (DoS)	API jadi lambat atau down	Trafik dibatasi dan diprioritaskan
API Abuse (Scraping, spam, dll)	Data diambil massal	Batasi jumlah data diambil per waktu

Rate Limiting & Throttling – (4/6)

Strategi Rate Limitting

Strategi	Deskripsi	Contoh
Fixed Window	Hitung jumlah request dalam interval tetap	100 request per menit
Sliding Window	Interval bergerak (lebih adil)	100 request per 60 detik dari waktu pertama request
Token Bucket	Tiap request butuh “token”; token diisi ulang periodik	Cocok untuk trafik fluktuatif
Leaky Bucket	Request diproses konstan, sisanya antri atau dibuang	Cocok untuk throttling rate stabil

Rate Limiting & Throttling – (5/6)

Best Practice

- Gunakan rate limit **per user, IP, atau API key**
- Gunakan **Redis** atau cache terdistribusi untuk melacak limit di banyak server
  Tampilkan **header informasi rate limit**:
  X-RateLimit-Limit: 100
  X-RateLimit-Remaining: 20
  X-RateLimit-Reset: 1696155600
- Beri **error code standar**: 429 Too Many Requests
- Logging untuk deteksi abuse pattern
- Gunakan kombinasi dengan **Authentication & Authorization**

Rate Limiting & Throttling – (6/6)

SGI Asia

OWASP Mapping:

- API4 – Unrestricted Resource Consumption

07.

LOGGING & MONITORING API ACTIVITY

Logging & Monitoring API Activity – (1/8)

SGI Asia

Apa itu logging?

Logging adalah proses **mencatat aktivitas atau kejadian** yang terjadi di sistem/API, seperti:

- siapa yang mengakses (user/IP/API key)
- endpoint yang diakses
- waktu, status code, dan hasilnya
- error atau exception yang terjadi

Tujuan utamanya: **mendokumentasikan jejak aktivitas API** agar bisa dianalisis.

Logging & Monitoring API Activity – (2/8)

Apa itu monitoring?

Monitoring adalah proses **mengamati dan menganalisis log secara real-time** untuk:

- mendeteksi serangan
- melihat performa sistem
- memastikan SLA dan uptime

Tujuannya: **deteksi dini masalah & ancaman** sebelum merusak sistem.

Logging & Monitoring API Activity – (3/8)

Mengapa Logging & Monitoring Penting di API Security?

Risiko tanpa Logging	Dampak	Manfaat dengan Logging & Monitoring
Serangan tidak terdeteksi	Tidak tahu kapan & dari mana datangnya	Dapat melacak sumber dan pola serangan
Kesalahan tidak terekam	Sulit troubleshooting	Debugging cepat dan efisien
Aktivitas ilegal tidak ada jejak	Tidak bisa audit forensik	Bisa audit aktivitas user/API key
API lambat tapi tidak tahu sebabnya	Tidak terpantau performa	Bisa analisis bottleneck dan response time

Logging & Monitoring API Activity – (4/8)

Monitoring Tools yang Umum Digunakan

Kategori	Contoh Tools	Fungsi
Log Management	ELK Stack (Elasticsearch, Logstash, Kibana), Loki + Grafana	Analisis dan visualisasi log
Monitoring & Alerting	Prometheus + Grafana, Datadog, New Relic	Mendeteksi anomali & mengirim alert
Cloud-native	AWS CloudWatch, Azure Monitor, GCP Stackdriver	Observabilitas sistem API di cloud

Logging & Monitoring API Activity – (5/8)

Monitoring Tools yang Umum Digunakan

- ✓ **Log semua aktivitas penting**, termasuk:
  - Request (URL, metode, IP, user)
  - Response (status code, waktu eksekusi)
  - Error, exception, atau timeout

Logging & Monitoring API Activity – (6/8)

Monitoring Tools yang Umum Digunakan

- ✓ **Hindari menyimpan data sensitif** di log:
  - Jangan log password, token, atau data pribadi (PII)
  - Gunakan masking/enkripsi untuk data sensitif
- ✓ **Gunakan format terstruktur (JSON)** agar mudah diproses oleh tools monitoring
- ✓ **Gunakan correlation ID / trace ID** untuk melacak request antar microservices
- ✓ **Gunakan alert otomatis** jika ada anomali (misal banyak error 401/500)

Logging & Monitoring API Activity – (7/8)

Contoh Log Ideal

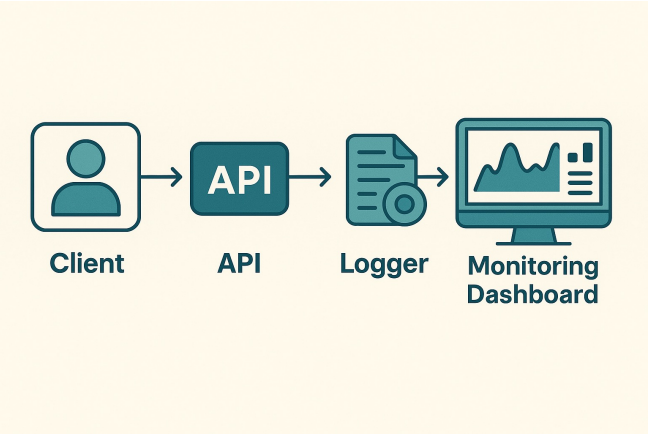
```
{
  "timestamp": "2025-10-14T13:00:00Z",
  "level": "INFO",
  "user_ip": "192.168.1.10",
  "endpoint": "/api/login",
  "method": "POST",
  "status": 200,
  "response_time_ms": 120,
  "trace_id": "abc123xyz"
}
```

Logging & Monitoring API Activity – (8/8)

Kesimpulan

Tanpa Logging
Tidak tahu siapa dan kapan API diserang

Dengan logging & Monitoring
Semua aktivitas tercatat → serangan & error mudah dilacak



Logging & Monitoring memastikan setiap aktivitas API terekam dan terpantau, sehingga tim bisa mendeteksi, menelusuri, dan merespons insiden keamanan dengan cepat.

Logging & Monitoring API Activity – (8/8)

OWASP Mapping:

Aspek	OWASP API Category	Fokus Risiko	Relevansi Logging & Monitoring
Configuration & Audit Trail	**API8: Security Misconfiguration**	Log tidak aktif / salah konfigurasi	Tidak bisa mendeteksi serangan atau perubahan penting
Third-party API usage	**API10: Unsafe Consumption of APIs**	Konsumsi API eksternal tanpa pengawasan	Tidak ada visibilitas & alert pada API-to-API calls

08.

WRITING EFFECTIVE PENETRATION TESTING REPORT

WRITING EFFECTIVE PENETRATION TESTING REPORT – (1/8)

Tujuan Utama Penetration Testing Report

Tujuan laporan pentest bukan sekadar menunjukkan hasil eksploitasi, tapi:

- Memberikan pemahaman risiko keamanan secara bisnis dan teknis
- Membantu tim developer memperbaiki celah keamanan
- Menyediakan dokumentasi resmi untuk audit dan compliance

Jadi, laporan yang efektif = **teknis** + **komunikatif** + **actionable**.

WRITING EFFECTIVE PENETRATION TESTING REPORT – (2/8)

Struktur Laporan Pentest

A. Executive Summary (Non-teknis)

- Ditujukan untuk manajemen / stakeholder non-teknis.
- Gunakan bahasa yang sederhana dan ringkas.

Isi umumnya:

- Tujuan pengujian
- Ruang lingkup (scope)
- Metodologi singkat
- Ringkasan hasil: jumlah temuan, level risiko (High/Medium/Low)
- Dampak terhadap bisnis
- Rekomendasi umum

WRITING EFFECTIVE PENETRATION TESTING REPORT – (3/8)

Struktur Laporan Pentest

A. Executive Summary (Non-teknis)

Contoh:

Pengujian menemukan 3 celah berisiko tinggi yang memungkinkan pencurian data pengguna. Kami merekomendasikan penerapan rate limiting dan validasi input di sisi server untuk mencegah eksploitasi lanjutan.

WRITING EFFECTIVE PENETRATION TESTING REPORT – (4/8)

Struktur Laporan Pentest

B. Technical Findings

Bagian	Deskripsi
ID Temuan & Judul	Contoh: API-001 – SQL Injection in /api/user/details
Severity	High / Medium / Low (berdasarkan CVSS atau risk rating internal)
Deskripsi	Jelaskan kerentanannya dan bagaimana bisa terjadi
Evidence / Proof of Concept	Tampilkan request/response, kode, atau screenshot
Dampak	Apa yang bisa dilakukan attacker
Langkah Reproduksi (Step to Reproduce)	Step-by-step cara eksploitasi
Rekomendasi / Remediasi	Cara memperbaiki, misalnya: gunakan parameterized query
Referensi	OWASP, CWE, NIST, atau best practice terkait

WRITING EFFECTIVE PENETRATION TESTING REPORT – (5/8)

Prinsip Penulisan Efektif

Prinsip	Penjelasan	Contoh
Clarity (Jelas)	Hindari jargon berlebihan, jelaskan dalam konteks API	"Endpoint /api/login rentan brute force karena tidak menerapkan rate limiting."
Accuracy (Akurat)	Pastikan bukti dan hasil eksploitasi bisa direproduksi	Lampirkan curl request dan respons server
Actionable (Dapat Ditindaklanjuti)	Setiap temuan harus punya solusi praktis	"Gunakan bcrypt untuk hashing password."
Visual Support	Gunakan diagram, tabel, dan screenshot untuk memperjelas	Screenshot hasil Burp Suite, flow request API
Risk Mapping	Hubungkan ke OWASP API Top 10 / CWE	"CWE-89: SQL Injection – API1: Broken Object Level Authorization"

WRITING EFFECTIVE PENETRATION TESTING REPORT – (6/8)

Contoh Template Sederhana

- Lihat lampiran Example_report.docx

WRITING EFFECTIVE PENETRATION TESTING REPORT – (7/8)

Tool Pendukung Penulisan Laporan

Kategori	Tools	Fungsi
Dokumentasi	Markdown, Word, Notion	Penulisan laporan
Screenshot & Proofs	Burp Suite, Postman, Fiddler	Bukti eksploitasi
Template Generator	Dradis, Serpico, Faraday	Standarisasi laporan
CVSS Calculator	NVD.NIST.gov	Menentukan tingkat keparahan (Severity)

WRITING EFFECTIVE PENETRATION TESTING REPORT – (8/8)

Best Practices

- Gunakan **bahasa profesional dan netral**
- Hindari **terlalu teknis di executive summary**
- Kelompokkan temuan berdasarkan risiko (High → Low)
- Tutup laporan dengan **section rekomendasi umum dan mitigasi prioritas**

SEKIAN TERIMA KASIH